

Allowed Callback URLs

https://github-user-search-app.netlify.app, http://localhost:3000

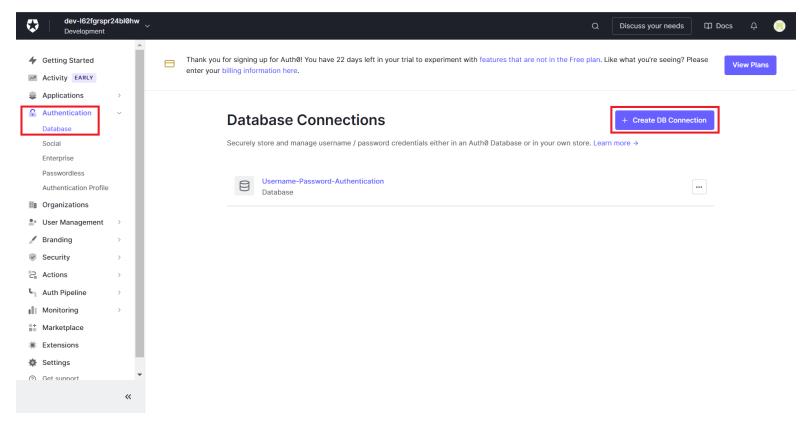
up lên host rồi thì cũng co thể dùng 2 domain cũng đc

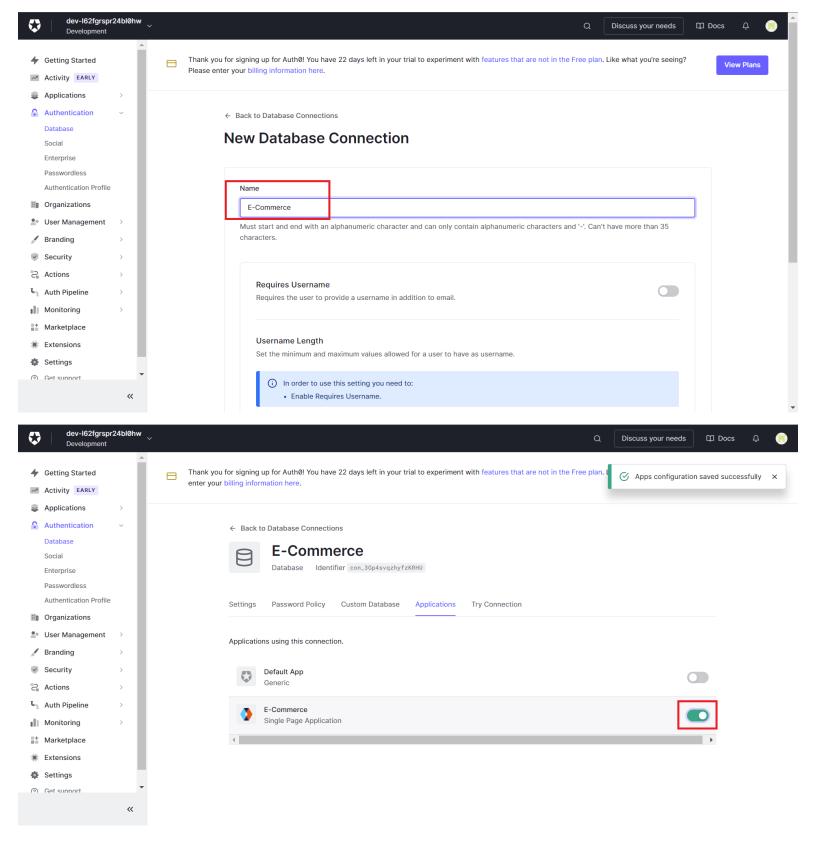
After the user authenticates we will only call back to any of these URLs. You can specify multiple valid URLs by comma-separating them (typically to handle different environments like QA or testing). Make sure to specify the protocol (https://) otherwise the callback may fail in some cases. With the exception of custom URI schemes for native clients, all callbacks should use protocol https://.

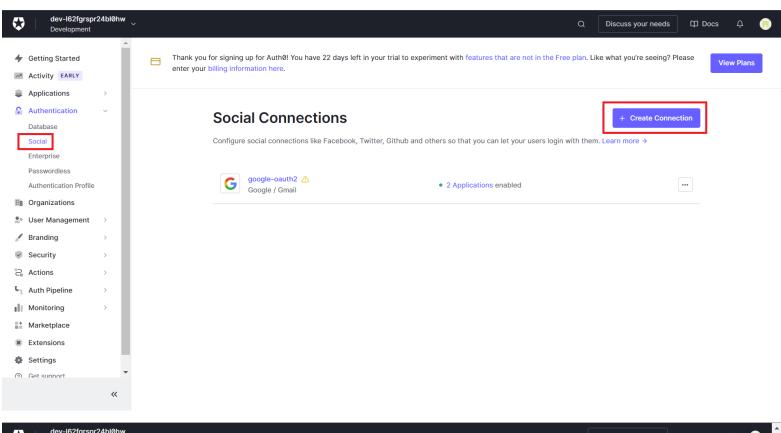
Allowed Logout URLs

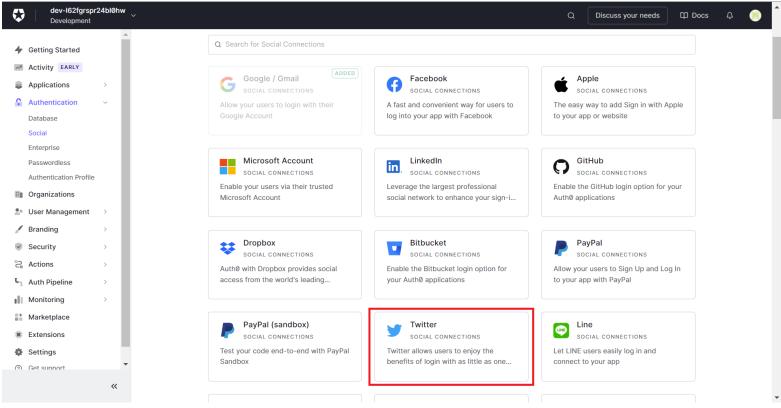
https://github-user-search-app.netlify.app, http://localhost:3000

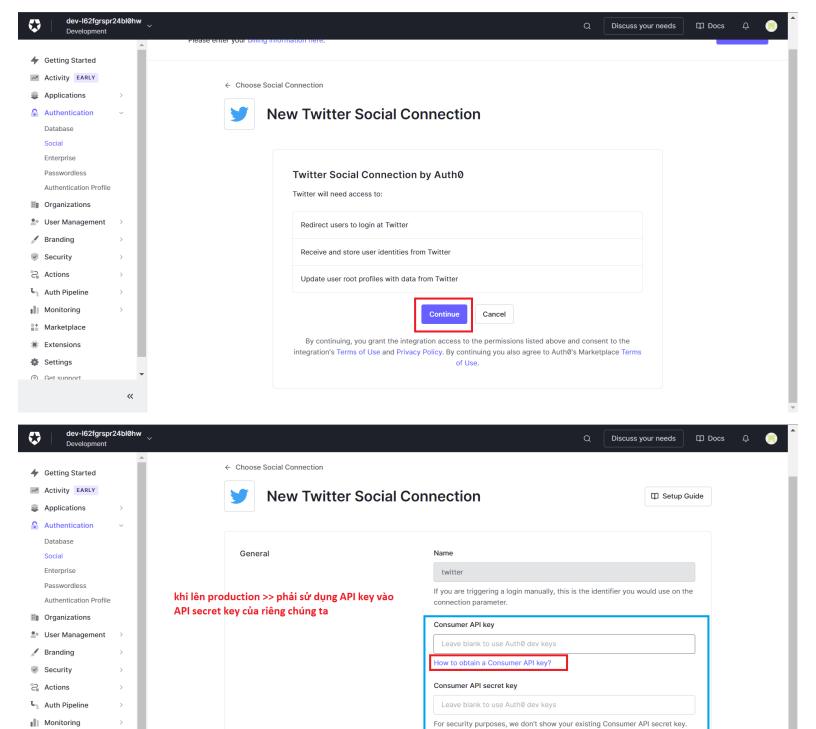
A set of URLs that are valid to redirect to after logout from Auth0. After a user logs out from Auth0 vou can redirect them with the returnTo











Basic Profile ③ REQUIRED

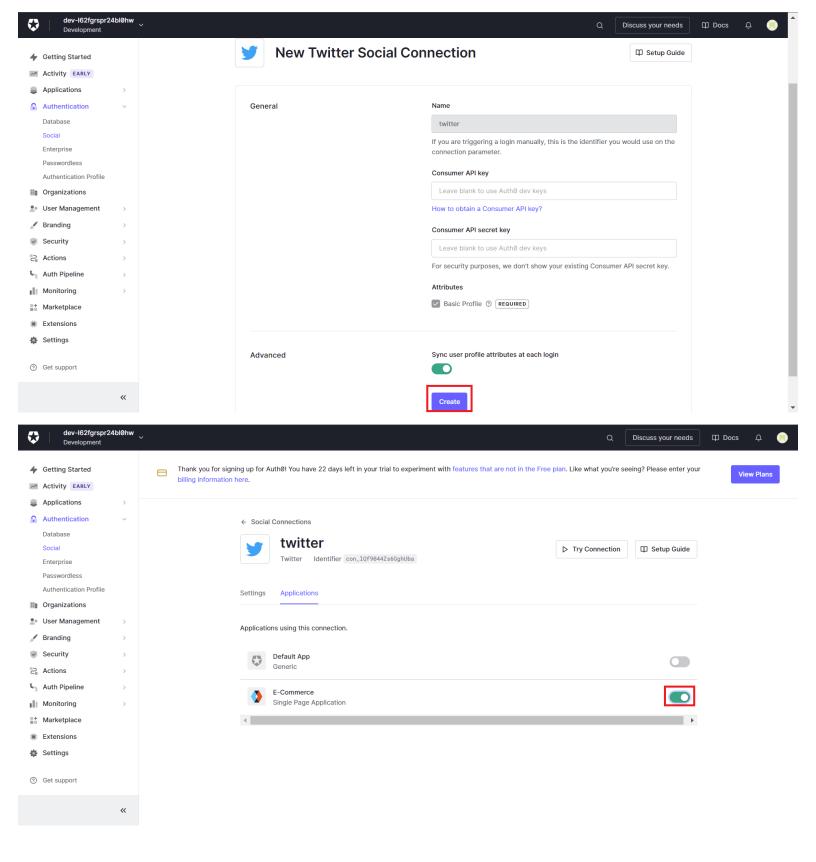
Sync user profile attributes at each login

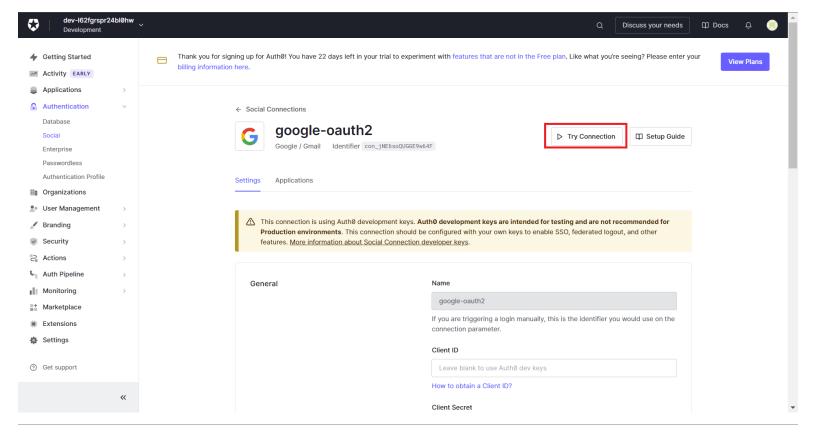
MarketplaceExtensions

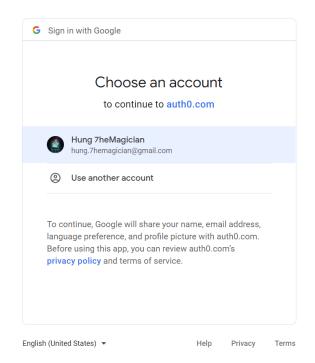
Settings

Get support

Advanced







nếu login thành công thì sẽ trả về như thế này

If you can see this page, it means that your connection works.

This is the user profile the application will receive:

```
{
  "sub": "google-oauth2|110025094973601652221",
  "given_name": "Hung",
  "family_name": "7heMagician",
  "nickname": "hung.7hemagician",
  "name": "Hung 7heMagician",
  "picture": "https://lh3.googleusercontent.com/a/AEdFTp5AqvX4jo6f3n65Pf7iZ4JaDfBjetCc4zRQHfyM=s96-c",
  "locale": "en",
  "updated_at": "2022-12-14T04:50:13.763Z"
}
```

Install the Auth0 React SDK ~



Configure the AuthOProvider component

For the SDK to function properly, you must set the following properties in the Auth0Provider component:

- domain: The domain of your Auth0 tenant. Generally, you can find this
 in the Auth0 Dashboard under your Application's Settings in the Domain
 field. If you are using a custom domain ☑, you should set this to the
 value of your custom domain instead.
- clientId: The ID of the Auth0 Application you set up earlier in this
 quickstart. You can find this in the Auth0 Dashboard under your
 Application's Settings in the Client ID field.
- redirecturi: The URL in your application that you would like Auth0 to redirect users to after they have authenticated. This corresponds to the callback URL you set up earlier in this quickstart. You can also find this

```
Log In & Download Sample
                        View on Github
                                                      Copy
Log In to configure with your own application.
            index.js
                      login.js logout.js
                                           profile.js
     import { Auth@Provider } from "@auth@/auth@-react";
     import React from "react";
     import ReactDOM from "react-dom";
     import App from "./App";
     ReactDOM.render(
       <Auth@Provider
         domain="YOUR_DOMAIN"
         clientId="YOUR_CLIENT_ID"
         redirectUri={window.location.origin}
         <App />
       </Auth@Provider>,
       document.getElementById("root")
```

Docs

```
import React from 'react';
    import { useAuth0 } from '@auth0/auth0-react';
    function Wrapper({ children }) {
      const {
        isLoading,
        error,
      } = useAuth0();
      if (isLoading) {
        return <div>Loading...</div>;
10
      if (error) {
11
        return <div>Oops... {error.message}</div>;
12
13
14
      return <>{children}</>;
15
    export default Wrapper;
16
```

isLoading vs error

Protect a route

Protect a route component using the withAuthenticationRequired higher order component. Visits to this route when unauthenticated will redirect the user to the login page and back to this page after login.

```
import React from 'react';
import { withAuthenticationRequired } from '@auth0/auth0-react';

const PrivateRoute = () => (<div>Private</div>);

export default withAuthenticationRequired(PrivateRoute, {
    // Show a message while the user waits to be redirected to the login page.
    onRedirecting: () => (<div>Redirecting you to the login page...</div>)
});
```

Protect a route